**II (районний) етап Всеукраїнської учнівської олімпіади з інформатики  
Київ, 19 листопада 2017 р.**

Максимальна оцінка за кожну з чотирьох задач — 100 балів.

Для всіх задач обмеження на час — 1 секунда / тест; обмеження на пам’ять — 256 МБ.

Матеріали олімпіади буде оприлюднено на сайті [**kievoi.ippo.kubg.edu.ua**](http://www.kievoi.ippo.kubg.edu.ua), а також на [**soi.org.ua**](http://soi.org.ua/).  
Автор задач — Данило Мисак.

**1. Процесор** (назва програми: **cpu.cpp** / **cpu.pas** / **cpu.\***)

Основним компонентом кожного комп’ютера є процесор — чип, що безпосередньо виконує усі операції, які потрібні для роботи програми. Уявіть, що вашій програмі необхідно виконати рівно операцій, а проце­сор працює зі швидкістю операцій за секунду. Підрахуйте, за скільки повних се­кунд про­цесор зможе ви­конати всі операції.

**Вхідні дані**

У єдиному рядку вхідного файлу задано натуральні числа та , причому обидва числа менші за .

**Вихідні дані**

У вихідний файл виведіть кількість повних секунд, після яких виконання операцій буде завершеним.

**Приклади**

|  |  |
| --- | --- |
| Вхідний файл **cpu.in** | Вихідний файл **cpu.out** |
| 35 5 | 7 |
| 52 7 | 8 |

**Коментарі до прикладів**

У першому прикладі за секунд процесор виконає рівно операцій. У другому прикладі за се­кунд процесор виконає операцій, а виконання решти трьох операцій завершить упродовж вось­мої секунди.

**Історична довідка**

Перші універсальні процесори на кшталт сучасних почали з’являтися у —-х роках завдяки зусиллям американських та японських інженерів. Комп’ютери існу­вали й до того, але кожна модель могла виконува­ти лише свій унікальний набір інструкцій, спрямованих на вирішення конкретного ко­ла задач.

**2. Миша** (назва програми: **mouse.cpp** / **mouse.pas** / **mouse.\***)

Одним із базових засобів для взаємодії з комп’ютером є миша та подібні до неї пристрої. Найчастіше миші мають по дві кнопки — ліву та праву, — а різноманітні дії можна виконати за допомогою таких операцій, як затис­кан­ня/від­пус­кання лівої кнопки, затискання/відпускання правої кнопки або ж подвійне натискання лівої кнопки миші. Визна­чимо ос­танню операцію формально: подвійним натисканням назвемо послідов­ність із трьох дій — за­тис­кання лівої кноп­ки, відпускання лівої кнопки та ще одного затискання лівої кнопки миші — за умови, що всі три дії було виконано протягом півсекунди, вони не переривалися іншими опера­ціями з мишею (тоб­то затис­канням правої кноп­ки), а на момент перед виконанням даних трьох дій жодна з кнопок миші не була затисне­на.

Ваше завдання — за послідовністю дій із мишею встановити, скільки разів було здійснено подвійне натис­кання лівої кнопки.

**Вхідні дані**

У першому рядку вхідного файлу вказано кількість дій ; вона є натуральним числом та не перевищує . У наступних ряд­ках за­дано самі дії:

1. Перше число — тип дії: для затискання лівої кнопки, для відпускання лівої кнопки; для затис­кання правої кнопки, для відпускання правої кнопки.
2. Друге число — часова мітка дії — момент часу в секундах, у який було зроблено дію. Часова мітка є додат­ним числом, і для нього вказано рівно три цифри після крапки, яка розділяє цілу та дробову ча­стини числа.

Дії задано в порядку зростання часу, причому жодні дві часові мітки не збігаються. Початково (у нульовий момент часу) жодна з кно­пок миші не затиснена. Остання часова мітка у файлі менша за секунд. Дії задано коректно: наприклад, між двома діями затискання однієї й тієї ж кнопки обов’язково знайдеться дія її від­пус­кання, а між двома діями відпускання однієї кнопки — дія її затискання.

**Вихідні дані**

У вихідний файл виведіть кількість подвійних натискань лівої кнопки миші.

**Приклади**

|  |  |
| --- | --- |
| Вхідний файл **mouse.in** | Вихідний файл **mouse.out** |
| 7  1 0.500  2 0.994  1 1.000  2 5.000  1 8.500  2 8.994  1 9.001 | 1 |
| 15  1 2.345  2 2.346  1 2.347  2 2.348  1 2.349  2 2.350  1 12.345  2 12.346  3 12.347  1 12.348  2 12.349  4 12.350  1 22.345  2 22.346  1 22.347 | 2 |

**Коментарі до прикладів**

У першому прикладі трійка дій між мітками та не є подвійним натисканням, бо виконується біль­ше ніж півсекунди. У другому прикладі трійка дій між мітками та не є подвійним натискан­ням, бо затискання лівої кнопки на є фінальною дією попереднього подвійного натискання.

**Історична довідка**

Мишу було винайдено у середині -х років. Спершу її винахідник, американець Дуґлас Ен­ґельбарт, на­зи­вав свій пристрій «жуком», але дріт, який був приєднаний до пристрою з заднього боку, однозначно нага­дував мишачий хвіст і змусив інженера змінити думку.

**3. Клавіатура** (назва програми: **keyboard.cpp** / **keyboard.pas** / **keyboard.\***)

Мабуть, основним засобом введення інформації в комп’ютер є клавіатура. Користувачу стає вкрай незруч­но, якщо клавіатура псується і час від часу не передає комп’ютеру інформацію про клавіші, які користувач натискає. На жаль, з однією клавіатурою трапилося саме таке: інколи клавіатура відключається та ігно­рує натискання клавіш; через одне або кілька натискань вона вмикається знову і працює в звичайному режимі.

Ваше завдання — за початковим текстом, який вводив користувач, та остаточним варіантом цього тексту, який опи­нився на комп’ютері, визначити, яку найменшу кількість разів клавіатура могла відключатися під час дру­ку.

**Вхідні дані**

У першому рядку вхідного файлу записано початковий текст, який вводив користувач: текст складається не більш ніж із малих літер латинської абетки та не містить жодних інших символів (зокрема й пробілів). У другому рядку за­писано текст, що в підсумку опинився на комп’ютері. Обидва тексти непорожні, і другий текст дійс­но міг ут­воритися з першого.

**Вихідні дані**

У вихідний файл виведіть ціле число — найменшу кількість разів, які могла відключатися клавіатура у про­цесі друку.

**Приклади**

|  |  |
| --- | --- |
| Вхідний файл **keyboard.in** | Вихідний файл **keyboard.out** |
| helloworld  hello | 1 |
| helloworld  lord | 3 |

**Історична довідка**

Клавіатура є одним із найстарших компонентів комп’ютера: її про­то­типами були телетайпи та навіть звичай­ні друкарські машинки, що існували вже два століття тому.

**4. Монітор** (назва програми: **monitor.cpp** / **monitor.pas** / **monitor.\***)

Перед монітором стоїть достатньо непроста задача: щосекунди по кілька десятків разів йому необхідно ви­водити зображення, що складається з величезної кількості пікселів по горизонталі та по вертикалі, а ко­жен піксель при цьому може бути зафарбований в один з мільйонів кольорів.

У цій задачі ви маєте сформувати лише один кадр, який виведе монітор. Екран монітора має прямокутний розмір. На екран потрібно в заданому порядку вивести фігури, які можуть частково або повністю перекри­вати одна одну: усі фігури — прямокутники однакового роз­міру та однакової орієнтації, і кожен прямокут­ник цілком зафарбовано в один і той самий колір.

**Вхідні дані**

У першому рядку вхідного файлу вказано ширину та висоту екрана в пікселях; обидва числа натуральні та не перевищують . У другому рядку вказано ширину та висоту у пікселях одного прямокутника; роз­міри прямокутника також є натуральними числами та не пере­вищують відповідних розмірів екрана. У тре­тьому рядку задано кількість прямокутників — натуральне число, не більше за . У наступних рядках містяться по три цілих числа, і кожна трійка за­дає відповідний прямокутник:

1. Перше число — відстань у пікселях від лівої межі екрана до лівої межі прямокутника.
2. Друге число — відстань у пікселях від верхньої межі екрана до верхньої межі прямокутника.
3. Третє число — колір усіх пікселів прямокутника — невід’ємне ціле число, менше за .

Усі прямокутників повністю вміщаються на екрані, а задано їх у такому порядку, що наступні прямокутни­ки, якщо накладаються на котрісь із попередніх, відображаються поверх них. Усі непокриті пікселі екрана, якщо такі є, мають колір .

**Вихідні дані**

У вихідний файл виведіть рядків по чисел у кожному так, щоб числа задавали кольори відповідних пік­селів на екрані монітора.

**Приклад**

|  |  |
| --- | --- |
| Вхідний файл **monitor.in** | Вихідний файл **monitor.out** |
| 5 3  3 2  4  0 0 5  1 1 6  1 0 0  0 1 3 | 5 0 0 0 0 3 3 3 0 0 3 3 3 6 0 |

**Історична довідка**

З англійської мови «монітор» — засіб спостереження. У перших комп’ютерах моніторами служили набори ламп, що почергово світилися і сигналізували таким чином про стан виконання програми. Зрештою інжене­ри зрозуміли, що телевізійний екран значно краще підходить для цих цілей.

**Ідеї розв’язання**

**1. Процесор**

Відповідь — округлена у більший бік частка від ділення на . Її можна отримати, наприклад, за формулою , де квадратні дужки позначають найбільше ціле число, що не перевищує дане. У мові Pascal цю формулу найпростіше записати так: (n+m-1) div m.

Крім то­го, слід врахувати, що двобайтових змінних (integer у мові Pascal) може не вистачити для збереження значень на .

**2. Миша**

Поступово зчитуючи рядки файлу, будемо запам’ятовувати три останніх дії, а також те, чи є на даний момент затисненою права кнопка миші. Якщо трьома останніми діями були затискання, відпускання і знову затискання лівої кнопки, між першим і другим затисканням пройшло не більше півсекунди, а права кнопка на даний момент не затиснена, додаємо одиницю до лічильника подвійних натискань. При цьому очищує­мо набір із трьох збережених дій, щоб помилково не зарахувати останнє затискання лівої кнопки за перше натискання в наступному подвійному.

Для роботи з часовими мітками достатньо використовувати змінні типу double.

**3. Клавіатура**

Задачу можна розв’язати за допомогою підходу, що називається *динамічним програмуванням*. Не­хай довжина початкового тексту — , а отриманого — . Через , де та — натуральні числа, що не перевищують та відповідно, позначимо мінімальну кількість відключень клавіатури після друку пер­ших символів тексту за умови, що перші символів початкового тексту перейшли в перші символів оста­точно­го тексту, причому -й символ перейшов у -й. Якщо такого бути не могло (наприклад, у випадку, коли -й символ початкового тексту та -й символ отриманого тексту різні), домовимося, що значення до­рівнює нескінченності, яку на практиці можна замінити просто дуже великим числом.

Через позначимо мінімальну кількість відключень після друку перших сим­во­лів тексту за умови, що перші символів початкового тексту перейшли в перші символів остаточно­го тексту, але -й символ не перейшов у -й, а випав. У випадку, коли цього статися не могло, аналогічним чином уважа­тиме­мо дане значення рівним нескінченності.

Відповіддю до задачі буде число , а щоб його підрахувати, ви­значатиме­мо значення величин та у порядку збільшення індексів та . У цьому допоможуть такі *рекурентні формули*:

(через і позначено -й символ початкового тексту та -й символ отриманого тексту відповідно). При цьому домовимося про такі початкові значення для або :

Складність алгоритму — .

**4. Монітор**

Задачу, звичайно, не так складно розв’язати, розглядаючи по порядку прямокутники й безпосеред­ньо присвоюючи усім пікселям, що їм належать, відповідні кольори. Це потребує часу та дозволяє набрати лише частину балів.

Побудуємо ефективніший розв’язок. Спершу розв’яжемо задачу для , тобто для випадку, ко­ли екран складається з одного ряду пікселів. Запишімо спершу в кожен піксель, що є першим (лівим) пік­селем принаймні одного з прямокутників, колір і порядковий номер цього прямокутника; якщо деякий піксель є лівим одразу для кількох прямокутників, запишемо в нього інформацію лише про той прямокут­ник, що має найбільший порядковий но­мер (він перекриє решту). Тепер будемо йти по екрану зліва напра­во, перебираючи його пікселі від першого до останнього і підтримуючи в кожен момент часу список прямо­кутників, які лежать під даним пік­селем, причому в порядку від найвищого прямокутника до найглибшого. Прямокутники, права частина яких ніколи не «визирне» з-під інших прямокутників набору, можна зі списку видаляти, адже на фінальне розфарбу­вання пікселів вони не вплинуть. Оновлювати даний список можна ефективно: перейшовши від попереднього пікселя до наступного, відкинемо зверху списку прямокутники, останнім пікселем яких був попередній; якщо в наступному пікселі починається новий прямокутник, то до­дамо цей прямокутник до списку ззаду, видаливши за по­треби з кінця списку усі ті прямокутники, що опи­няться під ним. У процесі проходу екрана дістанемо й від­повідь для кожного пікселя — це колір найвищо­го на да­ний момент прямокутника зі списку.

Розв’язок задачі для двовимірного випадку — коли екран складається більше ніж з одного рядка пікселів — полягає у комбінуванні наведеного вище підходу із самим собою. Фінальною частиною пов­ного алгоритму буде проходження окремо кожного рядка екрана зліва направо якраз у той спосіб, який на­ве­де­но вище. Але для того, щоб сформувати для кожного рядка масив з інформацією про те, який прямокутник почина­ється у кожному його пікселі, спершу застосуємо той самий підхід до стовпців: для кожного прямо­кутника запишемо його колір і порядковий номер у лівий верхній піксель даного прямокутника, після чого пройде­мося по кожному стовпцю зверху вниз і запишемо в кожен піксель екрана колір і порядковий номер найви­щого прямокутника, ліва межа якого проходить по даному пікселю.

Час виконання наведеного алгоритму є лінійним відносно сумарного розміру вхідних та вихідних даних; склад­ність алгоритму — .